Républiquealgérienne démocratique et populaire
Ministre de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – Biskra
Faculté des Sc.Exactes, et SNV.
Département d'Informatique

Formalisation de la procédure effective

Niveau: 1er année Master
Option: Génie Logiciel et Systèmes Distribués

2023/2024

Rappel

- Problème vs Algorithme
- Problème de décision
- Procédure effective
- Codage d un problème

Introduction

Exécution d'un programme sur un ordinateur ?

Un ordinateur qui exécute **un programme fixé**, est composé principalement de:

- Un processeur + un ensemble de registres dont il contient le compteur ordinal (CO).
- Une mémoire contenant les instructions du programme (ROM: car programme fixé).
- Une mémoire (RAM) contenant les données.

Exécution d'un programme sur un ordinateur?

L'exécution d'un programme sur un ordinateur est une succession de cycles contenant les étapes suivantes:

1. L'instruction indiquée par le CO est transféré de ROM vers le processeur (Registre instruction).
2. Le processeur exécute cette instruction et modifie, par conséquent, ses registres et/ou la mémoire RAM.
3. Le CO est incrémenté (ou modifié en cas de branchement).

Notion d'état

- D'une façon plus abstraite:

Si on connais le contenu des registres et de la mémoire des données,

➔ On peut connaitre sans ambiguïté, leurs contenus après l'exécution du cycle suivant.

- L'ensemble des informations nécessaires et suffisantes pour prédire son évolution future

➔ ETAT

Chaque contenu de la mémoire et des registres constitue un état différent.

NB: Il n'est pas nécessaire de connaître toutes la succession des instructions qui a abouti à l'état présent pour prédire l'état futur.

Fonction de transition

- Chaque cycle de l'exécution transforme l'état de la machine.
- Cette transformation ne dépend que du programme et de la machine.

Si le programme et la machine sont fixés

➔ transformation: état → état

= une fonction: États → États

- Elle est appelée une **fonction de transition.**
- Pour savoir ce qui va se passer durant l'exécution, il suffit de connaître:
  - La fonction de transition
  - L'état initial

Modélisation d'un programme exécuté par une machine

- Un programme exécuté par une machine est modélisé par:
  - Un ensemble ***fini*** d'états.
  - Une fonction de transition.
  - Un état initial.

- Une exécution du programme est représentée par la ***séquence des états*** obtenue par applications successives de la fonction de transition en partant de l'état initial.

Adaptation sur Les automates

- Automates finis : première modélisation de la notion de procédure effective.(Ont aussi d'autres applications).
- Dérivation de la notion d'automate fini de celle de programme exécutée sur un ordinateur : état, état initial, fonction de transition.
- Hypothèse du nombre d'états fini.

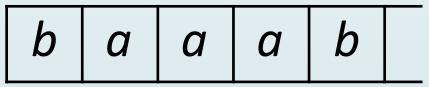
  → Conséquence : séquences d'états finies ou cycliques.
- Problème de la représentation des données : nombre de données différentes limitées car nombre d'états initiaux possibles fini.

Adaptation sur Les automates

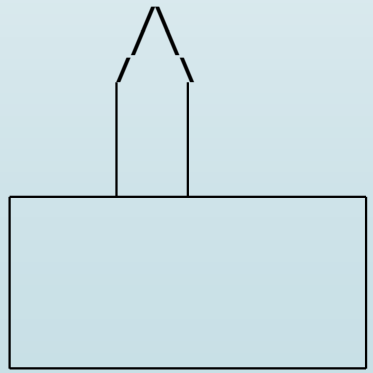
- Représentation des données
  1. Problème : reconnaître un langage.
  2. Données : mot.
  3. On supposera le mot fourni caractère par caractère, la machine traitant un caractère à chaque cycle et s'arrêtant à la fin du mot.

Description générale des automates

- Ruban d'entrée.
- Ensemble d'états :
  - état initial,
  - états accepteurs.
- Mécanisme d'exécution.
  - ruban :



  - Tête de lecture



Formalisation des automates fini

Un automate fini déterministe est défini par un quintuplait $M = (Q, \Sigma, \delta, s, F)$, où

1. Q est un ensemble fini d'états,
2. Σ est un alphabet,
3. $\delta : Q \times \Sigma \rightarrow Q$ est la fonction de transition,
4. $s \in Q$ est l'état initial,
5. $F \subseteq Q$ est l'ensemble des états accepteurs.

Langage accepté

- Configuration : $(q, w) \in Q \times \Sigma^*$.
- Configuration dérivable en une étape : $(q, w) \vdash_M (q', w')$.
- Configuration dérivable (en plusieurs étapes) : $(q, w) \vdash^*_M (q', w')$.
- Exécution d'un automate :

$$(s, w) \vdash (q_1, w_1) \vdash (q_2, w_2) \vdash \ldots \vdash (q_n, \varepsilon),$$

- Mot accepté :

$$(s, w) \vdash^*_M (q, \varepsilon) \text{ et } q \in F .$$

- Langage accepté $L(M)$:

$$\{ w \in \Sigma^* \mid (s, w) \vdash^*_M \quad (q, \varepsilon) \text{ avec } q \in F \}.$$

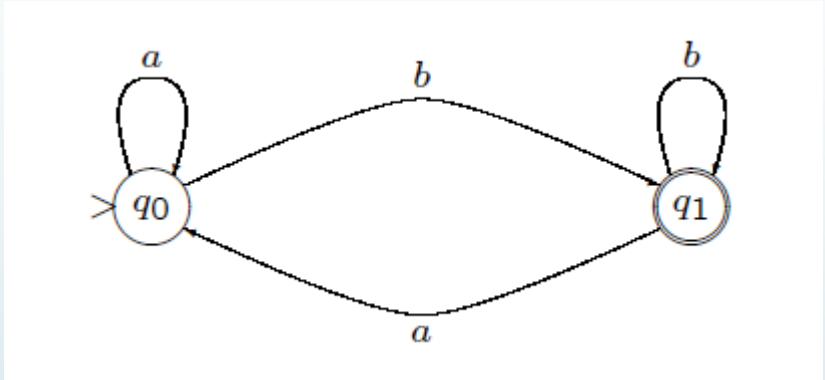
Exemple

- Mots se terminant par b :

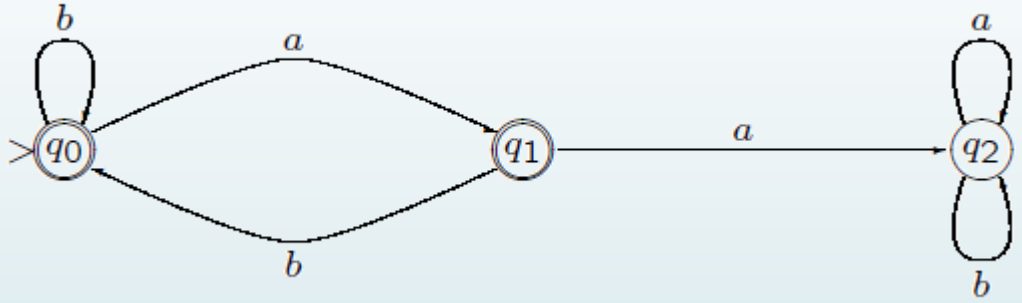
δ :

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_1

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$s = q_0$
$F = \{q_1\}$



Exemple



{w | w ne contient pas 2 a consécutifs}.

Les automates vs langage accepté

- Automates à états finis: ne peuvent pas reconnaitre les mots comme: $a^n b^n$.
- Automates à pile: ne peuvent pas reconnaitre les mots comme: $a^n b^n c^n$.
- Nous allons adopter les machines de Turing, car elles permettent de reconnaître tous les langages intuitivement reconnaissable par une procédure effective.